Practical Foundations of Debugging

Chapter 4

Instruction pointer

and disassembling a program with pointers

Designed, written and illustrated by Dmitry Vostokov

Instructions format

- inc mem/reg
- dec mem/reg

inc dword ptr [eax]

dec byte ptr [a]

- add mem/reg, reg/imm
- add reg, mem/imm

add byte ptr [eax], 10

add eax, dword ptr [a]

Logical shift instructions

- Shift to the left:

11111111 -> 11111110 ; shift by 1
11111110 -> 11110000 ; shift by 3
shl mem/reg, imm/reg
shl eax, 1
shl byte ptr [eax], ecx

- Shift to the right:

11111111 -> 01111111 ; shift by 1
01111111 -> 00001111 ; shift by 3
shr mem/reg, imm/reg
shr eax, 1
shr byte ptr [eax], ecx

Logical operations

- AND T = True F = False

```
1 and 1 = 1    T and T = T
1 and 0 = 0    T and F = F
0 and 1 = 0    F and T = F
0 and 0 = 0    F and F = F
```

- OR

```
1 and 1 = 1    T and T = T
1 and 0 = 1    T and F = T
0 and 1 = 1    F and T = T
0 and 0 = 0    F and F = F
```

Zeroing memory or registers

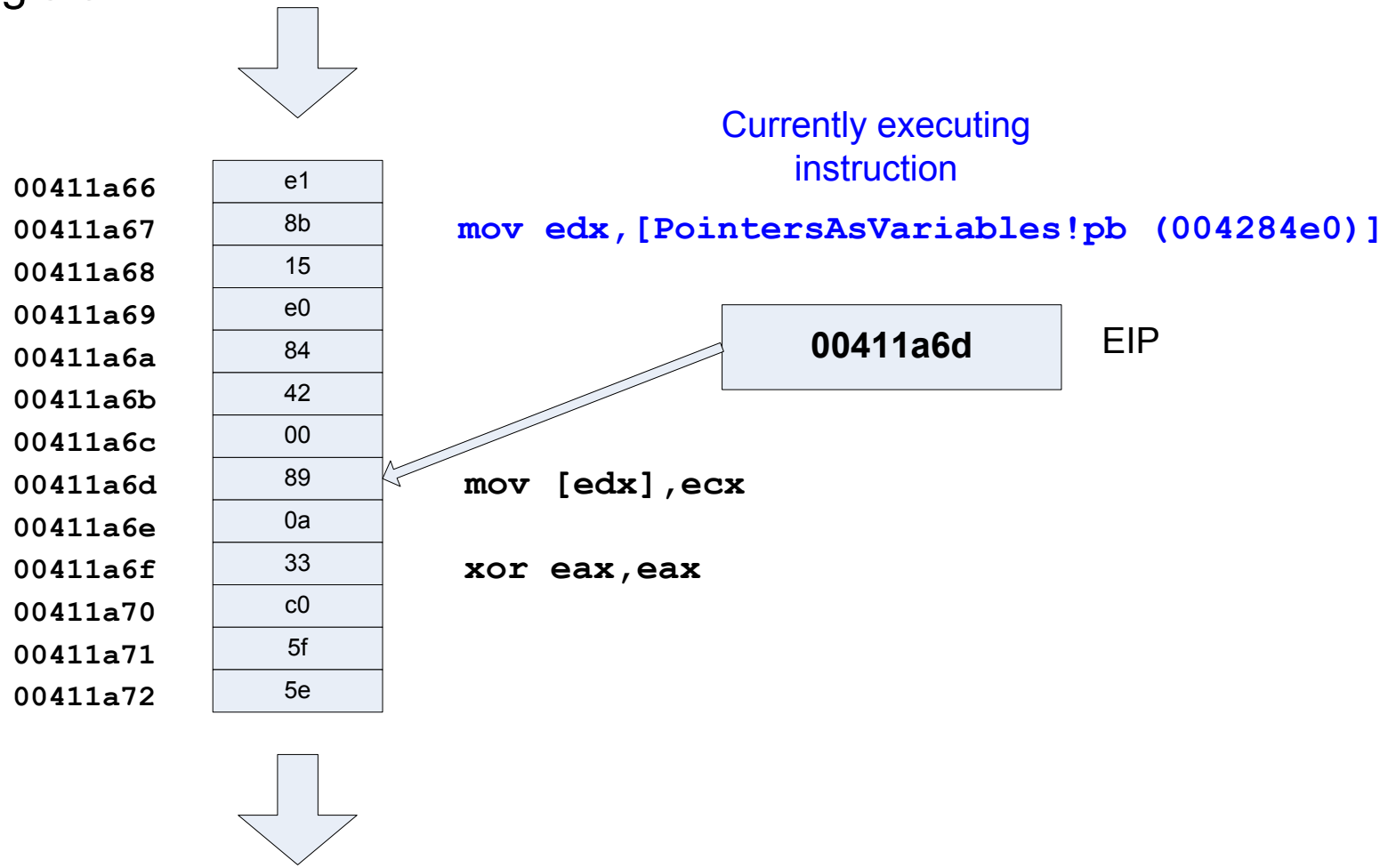
- mov dword ptr [a], 0
- mov eax, 0
- xor eax, eax

Exclusive OR

```
1 and 1 = 0    T and T = F
1 and 0 = 1    T and F = T
0 and 1 = 1    F and T = T
0 and 0 = 0    F and F = F
```

Instruction pointer

- Register EIP



Code memory segment

- .DATA
- .BSS
- .TEXT

Program code

Example of disassembly output – no optimizations

```
Memory   program code          disassembly
address


00411a1e c705e4844200ec844200 mov dword ptr [PointersAsVariables!pa (004284e4)],0x4284ec
00411a28 c705e0844200e8844200 mov dword ptr [PointersAsVariables!pb (004284e0)],0x4284e8
00411a32 a1e4844200           mov eax,[PointersAsVariables!pa (004284e4)]
00411a37 c70001000000         mov dword ptr [eax],0x1
00411a3d a1e0844200           mov eax,[PointersAsVariables!pb (004284e0)]
00411a42 c70001000000         mov dword ptr [eax],0x1
00411a48 a1e0844200           mov eax,[PointersAsVariables!pb (004284e0)]
00411a4d 8b08                 mov ecx,[eax]
00411a4f 8b15e4844200         mov edx,[PointersAsVariables!pa (004284e4)]
00411a55 030a                 add ecx,[edx]
00411a57 a1e0844200           mov eax,[PointersAsVariables!pb (004284e0)]
00411a5c 8908                 mov [eax],ecx
00411a5e a1e0844200           mov eax,[PointersAsVariables!pb (004284e0)]
00411a63 8b08                 mov ecx,[eax]
00411a65 d1e1                 shl ecx,1
00411a67 8b15e0844200         mov edx,[PointersAsVariables!pb (004284e0)]
00411a6d 890a                 mov [edx],ecx
00411a6f 33c0                 xor eax,eax
```

Reconstructing C code – Part 1

```
mov dword ptr [PointersAsVariables!pa (004284e4)],0x4284ec   ; [pa]  := address of a
mov dword ptr [PointersAsVariables!pb (004284e0)],0x4284e8   ; [pb]  := address of b
mov eax,[PointersAsVariables!pa (004284e4)]                  ; eax   := [pa]
mov dword ptr [eax],0x1                                      ; [eax] := 1
mov eax,[PointersAsVariables!pb (004284e0)]                  ; eax   := [pb]
mov dword ptr [eax],0x1                                     ; [eax] := 1
mov eax,[PointersAsVariables!pb (004284e0)]                  ; eax   := [pb]
mov ecx,[eax]                                                ; ecx   := [eax]
mov edx,[PointersAsVariables!pa (004284e4)]                  ; edx   := [pa]
add ecx,[edx]                                                ; ecx   := ecx + [edx]
mov eax,[PointersAsVariables!pb (004284e0)]                  ; eax   := [pb]
mov [eax],ecx                                                ; [eax] := ecx
mov eax,[PointersAsVariables!pb (004284e0)]                  ; eax   := [pb]
mov ecx,[eax]                                                ; ecx   := [eax]
shl ecx,1                                                    ; ecx   := ecx * 2
mov edx,[PointersAsVariables!pb (004284e0)]                  ; edx   := [pb]
mov [edx],ecx                                                ; [edx] := ecx
xor eax,eax                                                  ; eax   := 0
```

Reconstructing C code – Part 2

```
[pa]  := address of a          ;  int a, b; int *pa, *pb;
[pb]  := address of b          ;  pa = &a; pb = &b;
eax   := [pa]                  ;  *pa = 1;
[eax] := 1
eax   := [pb]                  ;  *pb = 1;
[eax] := 1
eax   := [pb]                  ;  ecx = *pb;
ecx   := [eax]
edx   := [pa]                  ;  ecx = ecx + *pa;
ecx   := ecx + [edx]
eax   := [pb]                  ;  *pb = ecx;
[eax] := ecx
eax   := [pb]                  ;  ecx = *pb;
ecx   := [eax]
ecx   := ecx * 2               ;  ecx = ecx * 2;
edx   := [pb]                  ;  *pb = ecx;
[edx] := ecx
eax   := 0                     ; 0
```

Reconstructing C code – Part 3

```
int a, b;
int *pa, *pb;

pa = &a;
pb = &b;

*pa = 1;
*pb = 1;

ecx = *pb;
ecx = ecx + *pa;
*pb = ecx;                          ; *pb = *pb + *pa;


ecx = *pb;
ecx = ecx * 2;                      ; *pb = *pb * 2;
*pb = ecx;
```

Reconstructing C code – C program

```
int a, b;
int *pa, *pb;

pa = &a;
pb = &b;

*pa = 1;
*pb = 1;

*pb = *pb + *pa;
*pb = *pb * 2;
```

Example of disassembly output – optimized program

```
Memory   program code          disassembly
address

00401000 c705c4724000cc724000 mov dword ptr [PointersAsVariables!pa (004072c4)],0x4072cc
0040100a c705c0724000c8724000 mov dword ptr [PointersAsVariables!pb (004072c0)],0x4072c8
00401014 c705cc72400001000000 mov dword ptr [PointersAsVariables!a (004072cc)],0x1
0040101e c705c872400004000000 mov dword ptr [PointersAsVariables!b (004072c8)],0x4
00401028 33c0                 xor     eax,eax
0040102a c3                   ret
```

Question: Why VC++ compiler did not optimize away the first two instructions?

What’s next?

- Stack
- Stack manipulation instructions
- Local variables